Journal of New Research Approaches in Management and Accounting
ISSN:2588-4573
http://www.majournal.ir

فصلنامه رویکردهای پژوهشی نوین در مدیریت و حسابداری
سال پنجم، شماره ۵۹، بهار ۱۴۰۰- جلد هشتم، ص ۷۰-۵۶

هسته (کرنل) سیستم عامل لینوکس و عملکرد ماژول ها

فرشید صهبا [۱]
احد تفضلی [۲]
فرزانه فرازنده [۳]

تاریخ دریافت: ۱۴۰۰/۰۲/۲۵ تاریخ چاپ: ۱۴۰۰/۰۳/۰۵

چکیده

امروزه که اینترنت نقش به سزایی در تبادل اطلاعات دارد و همه در فکر کسب اخرین اطلاعات می باشند ما نیزباید سعی کنیم ازعلوم به روزحداکثراستفاده رانماییم. یکی از این علم ها، سیستم عامل Linux است. Linux که در اینجا منظور کرنل لینوکس است، پراستفاده‌ترین سورس کد اپن‌سورس نرم‌افزاری در تاریخ نرم‌افزارهای کامپیوتری است و هر جایی که فکرش را بکنید حضور دارد؛ در کامپیوتر، سرور، موبایل، تلویزیون، زیردریایی، هواپیما، اتومبیل‌های خودران و حتی در ایستگاه بین‌المللی فضایی؛ اما شاید کمتر کسی بداند که کرنل واقعاً چیست و طرز کار آن چگونه است؟ منظور از هسته‌ی سیستم عامل چیست و کرنل لینوکس شامل چه مواردی می‌شود؟ ماژول هسته لینوکس چیست؟ انواع ماژول کدامند؟ و چگونه به هسته وارد میشوند؟ همگی سوال هایی هستند که در این تحقیق مورد بررسی قرار خواهند گرفت.

کلید واژگان

سیستم عامل، کرنل، لینوکس، ماژول، کرنل لینوکس، هسته

۱- استادیار، گروه مهندسی فناوری اطلاعات، دانشگاه غیاث الدین جمشید کاشانی، آبیک، ایران. (f_sahba@yahoo.com)
۲- دانشجوی کارشناسی ارشد، گروه مهندسی فناوری اطلاعات، دانشگاه غیاث الدین جمشید کاشانی، آبیک، ایران.
۳- دانشجوی کارشناسی ارشد، گروه مهندسی فناوری اطلاعات، دانشگاه غیاث الدین جمشید کاشانی، آبیک، ایران.

مقدمه

Red Hat Linux که در ابتدا فقط به منظور ارائه خدمات شبکه به کار می رفت، امروزه توسط شرکت ها، افراد و سازمان های دولتی جهت کاهش هزینه ها، افزایش کارایی انجام کار، مورد استفاده قرارمی گیرد.

ده ها میلیون نفر د ر سرتاسر جهان در محل کار و منزل از این سیستم عامل استفاده می کنند لینوکس را بهشت برنامه نویسان می نامن د و به ان لقب زیباترین دستاوردهمکاری جمعی بشر را داده اند.

هر سیستم عاملی دارای هسته یا به زبان دیگر، Kernelاست. اگر کرنل وجود نداشته باشد کامپیوترها و البته وسایلی مثل گوشی و تبلت کار نمی‌کنند. هر کاربری معمولاً از اپلیکیشن و نرم‌افزارهای زیادی استفاده می‌کند و همیشه کرنل است که کارهای اساسی و پایه‌ای را انجام می‌دهد. در واقع کرنل واسطه‌ی بین سخت‌افزار کامپیوتر و نرم‌افزار است.

کرنل با استفاده از نرم‌افزار راه‌انداز یا به زبان دیگر Driver، با قطعات سخت‌افزاری در ارتباط است و به اصطلاح صحبت می‌کند. در برخی سیستم عامل‌ها، درایورها بخشی از کرنل هستند و در برخی دیگر درایور به صورت ماژول‌های کرنل، قابل نصب است.

هسته‌ی سیستم عامل در مدیریت منابع سخت‌افزاری نیز مشارکت جدی دارد و مواردی مثل مقدار حافظه‌ی RAM خالی را مدیریت می‌کند و زمانی که اپلیکیشن جدیدی می‌خواهد در حافظه قرار بگیرد، موقعیت صحیح توسط کرنل انتخاب می‌شود. کرنل میزان استفاده از پردازنده‌ی اصلی یا CPU را نیز بهینه‌سازی می‌کند تا فرآیندهای مختلف در سریع‌ترین زمان ممکن و به موازات هم قابل اجرا باشد.

کرنل لینوکس ماژولار است، به این صورت که عملکردهای اصلی در فایل کرنل هستند، در حالی که درایورهای اختصاری به صورت ماژول‌های جداگانه در مسیر /lib/modules ساخته می‌شوند.

به جای اینکه کرنل بخواهد از همه ماژول‌های خود استفاده کند، با شناسایی سخت‌افزار می‌تواند در استفاده یا لود کردن ماژول‌های مرتبط تنها بخش‌های مورد نیاز را به درون حافظه کرنل وارد کند. در این شرایط کرنل همچنان می‌تواند با دسترسی به هزاران فایل دیگر از طریق ماژول‌ها یکپارچه باقی بماند. این ویژگی زمانی اهمیت زیادی پیدا می‌کند که سیستم می‌تواند وابسته به تغییرات سخت‌افزاری عکس العمل مناسب و صحیح را نشان دهد.

روش تحقیق

کرنل چیست؟

Kernel هسته ی اصلی سیستم‌عامل است که منابع سیستم مانند پردازنده، حافظه و ... را به برنامه‌های دیگر اختصاص می‌دهد. کرنل را مانند زیرساخت و بنای اصلی ساختمان در نظر بگیرید؛ تمامی وسائل از زیرساخت‌های ساختمان به منظور استفاده صحیح بهره می‌برند. به عنوان مثال، برای استفاده از یخچال به مکانی ثابت و پریز برق نیاز داریم. آشپزخانه، پریز برق و سیم‌کشی ساختمان، جزئی از زیرساخت خانه هستند و منابع مورد نیاز مانند انرژی الکتریکی را در اختیار وسایل مربوطه قرار می‌دهند.

کرنل مفاهیم انتزاعی را برای برنامه‌نویسان فراهم می‌کند که می‌خواهند اپلیکیشن‌هایی برای پلتفرم مورد نظر خود توسعه دهند. این مفاهیم انتزاعی شامل ساده‌سازی انجام کارهای پیچیده است (به عنوان مثالی برای مفاهیم انتزاعی، ذخیره شدن

فایل روی هارددیسک را در نظر بگیرید؛ برای این کار، نیازی به دانستن جایگاه بلاک‌ها و کلاسترهای اشغال شده در هارد توسط فایل مورد نظر ندارید. ولی می‌دانید فایل مثلاً X روی درایو C ذخیره شده است).

کرنل تنها قسمت نرم‌افزاری یک سیستم‌عامل نیست که دربرگیرنده ی مفاهیم انتزاعی است، بلکه در واقع یکی از مهم‌ترین قسمت‌های آن است. درایورهای سخت‌افزاری بواسطه ی کرنل با دیگر بخش‌ها ارتباط برقرار می‌کنند (یا در واقع صحبت می‌کنند)، بنابراین کرنل نیازی ندارد نحوهٔ صحبت کردن تک‌تک قسمت‌های سخت‌افزارهایی که ساخته می‌شود را یاد بگیرد. این چیزی است که باعث می‌شود یک کرنل، به تنهایی روی بسیاری از برندها و مدل‌های مختلف سخت‌افزاری اجرا شود.

آشنایی با Kernel لینوکس

شاید از نظر علمی درست نباشد که بگوییم لینوکس یک سیستم عامل کامل است! در واقع مفهوم دقیق Linux، هسته‌ای است که اولین بار توسط Linus Torvalds منتشر شده است. تمام موارد دیگر و در واقع آنچه حین کار با توزیعات مختلف لینوکس روی مانیتور مشاهده می‌کنید، توسط پروژه‌های نرم‌افزار و برنامه‌نویسان متعدد تهیه شده است.

کرنل لینوکس در سال ۱۹۹۱ منتشر شده و در ابتدا قرار بود نام این پروژه، Freax که ترکیبی از Free و Freak و UNIX است، باشد اما نام Linux توسط یکی از همکاران پیشنهاد شد. در سال ۱۹۹۲ اولین نسخه‌ی لینوکس توسط Torvalds تحت لایسنس حق نشر GNU منتشر شد.

بخش اعظم دستاپ لینوکس محصول GUN Project است که تقریباً یک سیستم عامل دستاپی کامل را شکل داد. به همین علت است که گاهی به این سیستم عامل، گنو لینوکس گفته می‌شود.

دستاپ‌های رایگان و متن باز دیگری مثل FreeBCD نیز موجود است که از نظر ظاهری و عملکردی شبیه لینوکس به نظر می‌رسد چرا که در این دستاپ‌ها هم از نرم‌افزارهای GNU استفاده شده است.

با توجه به اینکه هسته‌ی لینوکس تحت لیسانس GNU منتشر شده، بسیاری از کمپانی‌های نرم‌افزاری نیازی به طراحی کرنل مشابه و رقیب نمی‌بینند و به جای این کار در توسعه‌ی هسته‌ی لینوکس مشارکت می‌کنند. البته مایکروسافت و اپل که طراح ویندوز و مک‌اواس هستند، روش متفاوتی را در پیش گرفته و هسته‌ی متفاوتی طراحی کرده‌اند.

در حال حاضر کرنل لینوکس یک پروژه‌ی عظیم با میلیون‌ها خط کد است. هزاران توسعه‌دهنده و هزاران کمپانی بزرگ در توسعه‌ی هسته‌ی لینوکس مشارکت دارند و لذا لینوکس یکی از مشهورترین و پرکاربردترین نرم‌افزارهای متن باز در طول تاریخ محسوب می‌شود.

کاربردهای کرنل لینوکس

با وجود اینکه لینوکس یک سیستم عامل دستاپی قدرتمند است، کرنل لینوکس کاربردهای متنوع و متفاوتی دارد. یکی از کاربردهای عمومی، سیستم عامل اندروید برای گوشی‌ها و تبلت‌ها است. در واقع اندروید برخلاف بسیاری از توزیعات محبوب لینوکس، سیستم عامل دستاپی یا مخصوص سرورها و غیره نیست بلکه برای استفاده‌ی عموم افراد طراحی شده است. نسخه‌های خاص و ساده‌تر اندروید برای استفاده در وسایل پوشیدنی مثل ساعت هوشمند نیز در حقیقت مبتنی بر هسته‌ی لینوکس عمل می‌کند.

کاربرد دیگر هسته‌ی لینوکس در ابرکامپیوترها است. در واقع اغلب ابرکامپیوترها و درصد زیادی از سرورها که در فضای اینترنت با آنها سروکار داریم، از کرنل لینوکس بهره می‌برند و نه Windows مایکروسافت یا macOS اپل.

کامپیوترهای بسیار کوچک و بوردهای توسعه‌ی محبوب زیادی از کرنل لینوکس بهره می‌برد. Raspberry Pi یکی از این موارد است که اندازه‌ای در حد یک کارت اعتباری دارد. قیمت رزبری پای فقط ۳۵ دلار است در حالی که می‌توان روی آن یک توزیع سبک و ساده‌ی لینوکس را نصب و اجرا کرد. با وجود هزاران نرم‌افزار و پروژه‌ی متن باز، طبعاً توسعه‌ی نرم‌افزارهای جدید و خاص، نسبتاً ساده است.

آشنایی با وظایف Linux Kernel

حال می‌خواهیم بدانیم کرنل لینوکس چه مسئولیت‌هایی دارد؛ به عبارت دیگر، چه مفاهیم انتزاعی را باید از کرنل هر سیستم‌عاملی توقع داشته باشیم تا برایمان فراهم کند که در ادامه با برخی از مهم‌ترین آن‌ها آشنا خواهید شد:

ذخیره‌سازی داده

- حافظه با دسترسی تصادفی (RAM) به منظور خواندن و نوشتن متغیرها و داده‌ها در حافظه
- حافظه دائمی به منظور خواندن و نوشتن فایل‌ها روی ابزارهای ذخیره‌سازی دائمی مثل هارددیسک
- فایل سیستم مجازی

دسترسی به شبکه به منظور ارسال و دریافت داده‌ها روی یک شبکه کامپیوتری

- Physical Media Agnostic (اترنت، وایرلس، LTE، دایل‌آپ) که به روشی گفته می‌شود که هیچ پیش‌فرضی از ساختار شبکه از قبل وجود ندارد و باعث انعطاف‌پذیری بیشتر در شبکه‌های نامنظم می‌شود. کاربرد آن هم در مسائل نرم‌افزاری و هم در سخت‌افزاری است. به عنوان مثال، سیستم‌عامل‌ها و ابزارهای مختلف مانند گوشی همراه، تبلت و لپ‌تاپ از طریق پروتکل وای-فای قابلیت اتصال به یکدیگر را دارند.
- Partially Protocol Agnostic

زمان‌بندی کارها

- اشتراک‌گذاری زمانی پردازنده
- لود بالانسینگ و اولویت‌بندی کارها

پروتکل ابزارها (USB, FireWire, Serial, Parallel)

- فلش یواس‌بی
- وب کم
- ماوس و کیبورد

امنیت

- صدور پِرمیشن برای کاربران و گروه‌های کاربری
- صدور پِرمیشن به منابع

کرنل لینوکس با فراهم کردن سرویس‌های ذکر شده به صورت انواع مختلف فراخوان‌های سیستمی، توسعهٔ برنامه‌ها را آسان‌تر می‌کند. بیایید به برخی از روش‌هایی که باعث ساده‌سازی توسعه و تولید بیشتر می‌شود نگاهی بیاندازیم.

ذخیره‌سازی اطلاعات

دو روش برای ذخیره‌سازی اطلاعات وجود دارد که عبارتند از ذخیره‌سازی موقت و ذخیره‌سازی دائمی. ذخیره‌سازی موقتی به همان RAM داره دارد؛ هر چیزی که داخل رم شده، لازم نیست به طور دائمی ذخیره شود. یک نمونه برای درک بهتر این موضوع وقتی است که مشغول گشت‌وگذار در اینترنت هستید؛ شما قصد ندارید هر صفحه‌ای که مشاهده می‌نمایید به طور دائم در کامپیوترتان ذخیره شود.

کرنل لینوکس به طور شفاف و صرف نظر از سخت‌افزار استفاده شده، امکان خواندن و نوشتن اطلاعات روی رم را فراهم می‌کند. اهمیتی ندارد که لینوکس‌تان را روی پردازندهٔ قدیمی اینتل i386 اجرا کنید یا جدیدترین مدل بر پایهٔ ARM روی تلفن‌های همراه اندرویدی؛ در عین حال اصلاً نیازی نیست تا از قابلیت سازگاری اطمینان حاصل کنید و دست به تغییر کد بزنید.

به خاطر داشته باشید که کرنل لینوکس قسمتی است که به منظور پشتیبانی از سخت‌افزارهای مختلف تغییر می‌کند و تغییرات کرنل است که امکان استفاده از اینترفیسی عمومی را با وجود عدم استفاده از سخت‌افزارهای یکپارچه فراهم می‌کند.

به علاوه، کرنل تمامی پروسه‌ها را در حافظهٔ مخصوص به خودش نگاه‌داری می‌کند بدین معنی که خود پروسه‌ها نیازی به دانستن این که چه بخشی از حافظه در اختیار آن‌ها است ندارند چرا که تمام حافظه‌ای که برایشان قابل رؤیت است به آن‌ها تعلق دارد. در واقع، این روش در تقسیم‌بندی حافظه، امنیت را بدون نیاز به دخالت توسعه‌دهندگان افزایش می‌دهد.

ذخیره‌سازی دائمی هم ذخیره بر روی هارددرایو یا حافظهٔ فلش است. به طور مشابه، در حافظه‌های دائمی کرنل تفاوت در برقراری ارتباط با SATA، PATA، SCSI، USB، M.2 و پروتکل‌های دیگر ذخیره‌سازی را پنهان می‌کند و به یک برنامهٔ واحد اجازه می‌دهد تا به نوشتن و خواندن فایل‌ها روی هر واسطی و با استفاده از هر پروتکل و فایل سیستم شناخته شده‌ای بدون هیچ تغییری در برنامه بپردازد و این قابلیت کرنل قدرت بی‌نظیری را هم در اختیار توسعه‌دهنده و هم کاربر قرار می‌دهد. این قضیه همچنین قابلیت استفاده ی مجدد از کدها و بهره‌وری توسعه‌دهنده را افزایش می‌دهد چرا که نیازی به کدهای مخصوص برای پیکربندی‌های مختلف نخواهد بود.

دسترسی به شبکه

وقتی صحبت از شبکه به میان می‌آید، قضیه کمی متفاوت می‌شود چرا که هر پروتکل، قالب‌بندی مخصوص به خود را دارا است؛ بنابراین نیاز به کدنویسی مخصوص برای پشتیبانی هر کدام از پروتکل‌های شبکه حس می‌شود (خوشبختانه فقط پروتکل‌های IPv4 و IPv6 به صورت عمومی استفاده می‌شوند که در غیر این صورت، کار برای توسعه‌دهندگان کرنل لینوکس بسیار دشوار می‌شد). البته بسیاری از پروتکل‌های دیگر مانند IPX، DECnet و AppleTalk در لینوکس ساپورت می‌شوند ولی استفاده و پشتیبانی از این‌ها در اپلیکیشن‌های جدید فایدهٔ چندانی ندارد.

مجدد بپردازیم به IPv4 و IPv6. این دو، ساختار آدرس‌دهی بسیار متفاوتی با یکدیگر دارند ولی این قضیه به همان میزان که باعث دردسر می‌شود، سودمند نیز می‌باشد. پروتکل مورد نیاز -برای تشخیص توسط کرنل- به سادگی باتوجه به آدرس IP مشخص می‌شود. به علاوه، کرنل پشتیبانی از TCP،UDP،SCTP و ICMP را فراهم می‌کند که هر کدام از طریق فراخوان‌های سیستمی قابل استفاده خواهند بود. مهم نیست که سیستم شما توسط پروتکل Ethernet به شبکه متصل شده یا LTE یا Dialup؛ فراخوان‌های سیستمی همچنان یکسان خواهند بود.

تصور کنید بسته به این که از وای‌فای استفاده می‌نمایید یا اترنت، به نسخه‌های مختلفی از کروم یا فایرفاکس نیاز داشتید؛ که پیاده‌سازی این روش برای توسعه‌دهندگان کاری طاقت‌فرسا می‌بود؛ اما این قابلیت کرنل لینوکس نیز مفهوم بسیار قدرتمندی است و انعطاپذیری قابل‌توجهی را فراهم می‌کند که باعث افزایش بهره‌وری توسعه‌دهندگان و راحتی کاربران می‌شود.

زمان‌بندی کارها

زمان‌بندی تَسک‌های مختلف موضوع بسیار پیچیده و مهمی در لینوکس است؛ بنابراین به مباحث الگوریتم‌های زمان‌بندی کاری نداریم و تنها به مسئولیت‌ها و وظایف کرنل می‌پردازیم و این که کرنل چگونه نوبت هر پروسه را برای استفاده از پردازنده مشخص می‌کند حتی اگر صدها پروسهٔ مختلف در آنِ واحد وجود داشته باشند.

پیش از پیدایش پردازنده‌های چند هسته‌ای، کامپیوترها در حقیقت می‌توانستند تنها یک تَسک (وظیفه یا کار) را در لحظه‌ای خاص انجام دهند. هر یک از پروسه‌ها، سهم زمانی یکسانی را برای این که به ترتیب مورد پردازش قرار بگیرند در اختیار داشتند ولی این کار آن‌قدر سریع انجام می‌شد که باعث ایجاد تصور غلطی دربارهٔ اجرای هم‌زمان پروسه‌ها صورت گرفته بود.

تا قبل از پردازنده‌های چندهسته‌ای، تولیدکنندگان کامپیوترها برای اجرای بیش از یک فرایند در لحظه، باید بیش از یک پردازنده را در مادربورد تعبیه می‌کردند. این کار هنوز هم انجام می‌شود ولی با پردازنده‌های چندهسته‌ای و با استفاده از مفهومی تحت عنوان Hyperthreading که باعث اجرای دو فرآیند به صورت هم‌زمان بر روی یک هسته در برخی مدل‌های پردازنده‌های اینتل می‌شود (این ویژگی باعث می‌شود سیستم‌ها بتوانند در آن واحد بیش از صدها Thread را هَندل کنند).

هر پروسه نیاز به زمانی برای استفاده از پردازنده دارد و کرنل چیزی است که اطمینان حاصل می‌کند هر کدام از پروسه‌ها طبق زمان‌بندی به نوبت خود خواهند رسید. گذشته از این، برخی پروسه‌ها نیاز به تاٌخیر دارند که ممکن است به علت انتظار برای انجام عملیات I/O و یا هر چیز دیگری باشد. حال، به جای اشغال پردازنده در هنگام انتظار، فرایند دیگری می‌تواند نوبت را گرفته و اجرا شود و فرآیند اصلی بعد از گذراندن زمان مورد نیاز می‌تواند برای اجرا بازگردانده شود؛ در نتیجه، این کار باعث افزایش کارایی کلی سیستم می‌شود.

در مجموع، زمان‌بندی کارها بدان معنی است که توسعه‌دهنده نیازی به نگرانی دربارهٔ اجرای فرآیندهای دیگر روی کامپیوتر ندارد و فقط باید نگران اجرای بدون نقص برنامهٌ خود باشد.

تعداد مفاهیم انتزاعی مرتبط با کرنل لینوکس فوق‌العاده زیاد هستند و امکان پوشش دادن تمامی این مفاهیم در این مقاله وجود نداشت؛ به هر حال امیدواریم که دیدی کلی نسبت به نحوهٔ عملکرد کرنل لینوکس پیدا کرده باشید. همچنین اگر علاقمند به شروع یادگیری لینوکس و سیستم‌عامل گنو/لینوکس هستید، می‌توانید به دورهٔ آموزش لینوکس در سکان آکادمی مراجعه نمایید.

آشنایی با نسخه های مختلف لینوکس

Turbolinux

این نسخه برای شرکتها خوب است و نمی تواند برای دوستان خانه نشین این دیار خوب باشد. این نسخه برنامه های اضافی نیز دارد که کار مدیریت سیستم ها را در شرکت های بزرگ کنترل می کند. نمونه های زیادی مانند این نسخه وجود دارد ولی این نسخه بهترین انها محسوب می شود

Slackware Linux

این نسخه اولین نسخه ای بود که توزیع شد و نصب ان بسیار مشکل است. برخی از کاربران حرفه ای این نسخه استفاده می کنند. این نسخه کمترین طرفدار را دارد و یادگیری ان نیز مشکل است؛ اما ویژگی های خاص خودش را دارد. از جمله پایداری و کیفیت بالای ان را می توان نام برد.

Lycoris

این نسخه از جمله کامل ترین نسخه های موجود در بازار است. نصب ان اسان بوده و در بیشتر کامپیوترهای خانگی کار می کندبرنامه های بسیاری ضمیمه این نسخه از لینوکس هست.

Caldera OpenLinux

این نسخه توسط شرکت caldera توزیع میشود. البته این نسخه هم دانلود میشود و هم قابل خریداری است. این شرکت نسخه های دیگری هم توزیع کرده ولی نسخه مذکور بهتراز بقیه است.

Red Hat

این نسخه یکی از معروف ترین نسخه های لینوکس است؛ و اخرین نسخه ان ۹ می باشد. از این سیستم شرکتهای بزرگ سخت افزاری نظیر IBM، Dell پشتیبانی می کند؛ و به همین خاطر معروف شده است.

Hewlett-Packard

این نسخه در سایت redhat.com به صورت رایگان توزیع می شود.

Linux-Mandarke

این نسخه جزء آسان ترین نسخه های توزیع شده است و می تواند بهترین نسخه برای کابران مبتدی می باشد. بیشتر کاربران ا ز این نسخه استفاده می کنند. این نسخه در اینترنت به صورت رایگان موجود می باشد.

البته linux نسخه های زیادی دارد که فقط تعدادی از انها معرفی شدند.

ماژول هسته لینوکس

ماژول، قطعه ی نرم افزاری در بخشی جدا از Core هسته هستند که هنگام فراخونی شدن، پیوند و فعال میشوند و یکسری عملیات تعریف شده ای رو انجام میدهند.

ماژولها تکه کدهایی هستند که در حین اجرای هسته لینوکس می توانند وارد آن شده و یا از آن خارج شوند. این تکه کدها عملکرد هسته را بدون نیاز به راه اندازی دوباره کامپیوتر توسعه می دهند.

به عنوان مثال یک نوع از ماژولها device driver ها هستند که به هسته امکان استفاده از قابلیت سخت افزار ها را می دهند.

اگر ماژول ها وجود نداشتند، برای هر قابلیتی که می خواستیم به هسته اضافه کنیم یا از آن کم کنیم، می بایستی یک بار هسته را کامپایل می کردیم و برای استفاده از ان قابلیت یا حذف آن یک بار سیستم را از نو راه اندازی می کردیم.

این ماژول می توانند سرویس، filesystem، پروتکل شبکه، تعدادی System call و یا درایور یک سخت افزار باشند; که در هر صورت ماژول نام دارند.

ماژول ها در لینوکس به دو گروه تقسیم میشوند:

Kernel module

Loadable kernel module (یا LKM)

Kernel module: که با قرار دادن سورس ماژول در داخل پوشه های سورس Kernel، همراه با Kernel کامپایل میشود.

Loadable Kernel module: که با load کردن ماژول کامپایل شده در داخل سیستم در حال اجرا فعال میشود.

ماژول های گروه Loadable Kernel module (یا LKM) قطعه نرم افزاری هستند که می توانند در حین up بودن سیستم (بدون نیاز به reboot کردن)، Load و Unload شوند و یکسری عملیات تعریف شده ای را انجام بدهند.

برای درک بهتر این گروه از ماژول ها (LKM) میشه drive ها رو نام برد که موقع نصب فلش دیسک به سیستم، Load میشوند و موقع جداسازی فلش دیسک از سیستم، Unload میشوند؛ که البته میتوان ماژول های نوع LKM ای رو در boot سیستم هم قرار داد.

نکته اینکه این ماژول های LKM جزوی از kernel-space هستند و مستقیماً با base Kernel در تعامل هستند؛ و در زمان Load شدن ماژول، به قسمتی از وجود Kernel تبدیل میشوند.

همچنین بد نیست بدانید که از آغاز بوجود امدن هسته لینوکس کلاً چیزی بنام ماژول وجود نداشت و از نسخه ۱ به بعد این امکان فراهم شد.

نکته: چون ماژول ها در kernel-space توسعه داده میشوند، قادرند در انواع دیوایس های mobile،PC و embedded ها هم استفاده شوند.

همونطور که گفته شد برای استفاده از ماژول دو راه وجود دارد: یکی قرار دادن سورس ماژول در داخل پوشه های سورس Kernel و دومی Load کردن ماژول کامپایل شده در داخل سیستم در حال اجرا.

راه اولی به چندین دلیل پیشنهاد نمیشود:

در هر بار توسعه مجدد یا رفع باگ ماژول، باید کل Kernel رو مجدداً کامپایل و نصب کنید.

تشخصی خطا های منابع dependence شده بسیار مشکل هست.

کوچکترین باگی در ماژول باعث متوقف شدن boot موقع startup سیستم میشود; و باگ به سختی trace میشود.

از سرعت پایینتری برخوردار هست.

امکان ارسال پارامتر به ماژول وجود ندارد (باید reboot کنید و داستان...).

ماژول همیشه بی جهت و بدون استفاده Load میشود و باعث اشغال حافظه مجازی میشود.

در صورتی که راه دومی (یعنی ماژول نوع LKM) هیچ یک از این معایب رو ندارد و مزایای بیشتری هم دارد.

(البته شاید به دلایلی مجبور باشید از راه اول استفاده کنید که اون بحثش جداست)

موارد استفاده ماژول های LKM

ماژول های LKM در موارد بسیار زیادی مورد استفاده قرار می گیرند از جمله:

درایور سخت افزار: که واسط ارتباطات بین یک دستگاه (قطعه) خاص با سیستم عامل میشوند تا بتوانند با هم تعامل برقرار کنند. مثل کارت گرافیک، قفل های سخت افزاری، فلش دیسک و...

درایور filesystem: که مفسر نوع filesystem محتویات هارد دیسک (یا هر حافظه دیگه ای) هست. مثل FAT32 NTFS Ext4 SWAP و...

System calls: که شامل پیامها و دستورات built-in در Kernel هستند و برنامه های user-space(مثل gimp firefox openoffice و...) از اونها برای دریافت سرویس، callback و اطلاعات استفاده می کنند. مثل shutdown,readfile و...

درایور پروتکل شبکه: که مفسر و transformer اطلاعات رد و بدل شده در یک پروتوکل هستند. انها اطلاعات رو توسط stream ها در بین لایه های هسته جابه جا و تفسیر می کنند. مثل http,telnet,ssh,ftp و...

درایور پورت TTY: که واسط و تقویت کننده دستگاه هایی با رابط نوع ترمینال هستند (تصاویر). مثل بیسیم، فیش های هدفون، شارژر، ماشین های اداری، صنعتی...

و برنامه های کاربردی خاص که نیازمند توسعه در kernel-space هستند.

ماژول ها چگونه به هسته وارد می شوند؟

شما می توانید با اجرای دستور lsmod ماژول هایی که هم اکنون در هسته وارد شده اند را ببینید و از اطلاعات آنها باخبر شوید. این دستور اطلاعات خود را از فایل proc/modules/ دریافت می کند.

هنگامی که هسته، به امکان و عملکردی نیاز دارد که هم اکنون در آن نیست، یکی از deamon های آن به نام kmod دستور modprobe را اجرا می کند تا ماژول مربوطه که آن عملکرد را دارد وارد هسته شود. هنگامی که modprobe اجرا می شود به آن یک رشته کاراکتر به دو صورت زیر داده میشود:

۱) نام ماژول مانند softdog یا ppp

۲) یک مشخصه کلی مانند char-major-10-30

اگر حالت اول به modprobe داده شود، این دستور به دنبال فایلی به نام softdog.ko یا ppp.ko با روشی که در ادامه می آید می گردد.

ولی اگر حالت دوم به modprobe داده شود، این دستور ابتدا به دنبال رشته کاراکتر در فایل etc/modprobe.conf/ می گردد و اگر توانست alias یا مستعاری مانند:

Alias char-major-10-30 softdog پیدا کند، متوجه می شود که این نام کلی که در اینجا char-major-10-30 است به ماژول softdog اشاره می کند که فایل ماژول آن softdof.ko می باشد

در مرحله بعد modprobe فایل lib/modules/version/modules.dep/ را باز کرده و به دنبال ماژول هایی می گردد که باید قبل از ماژول مورد نظر به هسته وارد شوند. این فایل به وسیله دستور depmod -a ایجاد می شود و حاوی وابستگی بین ماژول هاست.

به عنوان مثال اگر به دنبال ماژول msdos.ko در این فایل بگردید خواهید دید که به ماژول دیگری به نام fat.ko وابسته است یعنی برای اینکه msdos.ko وارد هسته شود حتما باید قبل از ان fat.ko وارد شده باشد.

این مساله برای fat.ko نیز تکرار شده تا به مرحله ای برسیم که دیگر وابستگی موجود نباشد.

در نهایت modprobe دستور insmod را به کار می برد تا ابتدا وابستگی ها را به هسته وارد کرده و در نهایت ماژول مورد نظر ما به هسته وارد می شود.

پس modprobe وظیفه پیدا کردن ماژول، تعیین وابستگیهای آن و وارد کردن آن به هسته به وسیله صدا کردن insmod را دارد در حالی که insmod فقط وظیفه وارد کردن آن ماژول به هسته را دارد.

به عنوان مثال اگر بخواهیم به صورت دستی msdos.ko را وارد هسته کنیم به صورت زیر عمل می کنیم:

insmod /lib/modules/2.6.11/kernel/fs/fat/fat.ko #

معادل دو دستور بالا با modprobe به صورت زیر است:

insmod /lib/modules/2.6.11/kernel/fs/fat/msdos.ko #

مطلب قابل ذکر این است که insmod مسیر کامل تا فایل ماژول را می خواهد در حالی که modprobe فقط نام ماژول را می گیرد.

modprobe msdos #

قبل از شروع

قبل از اینکه وارد کد و کدزنی شویم چند نکته مهم را بررسی می کنیم:

۱-modversioning:

یک ماژول که برای یک هسته خاص کامپایل شده است بر روی هسته دیگر load نخواهد شد مگر اینکه شما CONFIG_MODVERSIONS را در هسته فعال کنید. در قسمت های بعد بیشتر به این مقوله خواهیم پرداخت.

۲- ماژولها نمی توانند چیزی به غیر از خطاها و هشدارها را بر روی صفحه نمایش نشان دهند. آنها برای نشان دادن اطلاعات خود، آنها را در log فایلها می نویسند

۳- مورد سوم که کاملا مورد قبول بنده نمی باشد این است که نویسنده می گوید:

اغلب توزیع کنندگان لینوکس کد منبع هسته را که مورد Patch نیز قرار گرفته به طرز غیر استانداردی توزیع می کنند که ممکن است باعث ایجاد مشکلاتی شود.

یکی از شایع ترین این مشکلات فایل های ناقص Header برای هسته لینوکس هستند. شما برای ماژول نویسی نیاز دارید که فایل های Header زیادی را در کدهای خود ضمیمه کنید و فایلهای ناقص اغلب فایل هایی هستند که برای ماژول نویسی به کار می روند." نویسنده پیشنهاد می کند که برای جلوگیری از این مشکل هسته را برای خود کامپایل کنید."

یک مثال – **ساده ترین ماژول**

برای شروع از مثال سنتی Hello World! شروع می کنیم. فایلی به نام hello-1.c باز کرده و کد C زیر را در آن بنویسید:

کد:

```
/* include <linux/module.h> /*needed by all modules#
/* include <linux/kernel.h> /*needed for Macros like KERN_INFO#
this function is called as initialization for all modules, if this function returns non- */
/*.zero means init_module failed and this module can't be loaded
{;int init_module(void) { printk(KERN_INFO "Hello World1.\n"); return 0
/* it iscalled when module is terminated and unloaded */
{;("void cleanup_module(void) { printk(KERN_INFO "Goodbye World1.\n
```

هر ماژول هسته ای حداقل بایستی ۲ تابع داشته باشد.

اولی تابع شروع که init_module() نامیده می شود و هنگام load شدن ماژول در هسته صدا زده می شود و دیگری تابع پایان که cleanup_module() نامیده می شود و هنگام unload شدن ماژول از هسته صدا زده می شود.

در قسمت های بعد به این موضوع می پردازیم که بعد از هسته ۲,۳,۱۳ شما می توانید هر نام دیگری برای این دو تابع قرار دهید.

با این حال خیلی از افراد هنوز از این استاندارد قدیمی استفاده می کنند. دو فایل Header در این کد ضمیمه شده اند. یکی linux/module.h می باشد که برای هر ماژولی مورد نیاز است و تعریف خیلی از توابع را در خود دارد و دیگری linux/kernel.h می باشد که حاوی تعدادی ماکرو می باشد مانند KERN_INFO.

مختصری درباره printk

بر خلاف آن چیزی که ممکن است درباره printk تصور کنید این تابع چیزی در صفحه نمایش چاپ نمی کند و برای کار با کاربر نیست. این تابع برای مکانیزم log هسته به کار میرود. هر printk با یک اولویت می آید که در این مثال ماکروی KERN_INFO برای این منظور به کار رفته است. تعداد ۸ اولویت وجود دارند که به صورت ماکرو در فایل

linux/kernel.h تعریف شده اند. اگر شما این اولویت را تعیین نکنید به طور پیش فرض DEFAULT_MESSAGE_LOGLEVEL به آن تخصیص می یابد.

اگر این اولویت کمتر از اولویت int console_loglevel (که در linux/kernel.h تعریف شده) باشد، message در دستور printk() بر روی صفحه ظاهر می شود. اگر syslogd و یا klogd در سیستم در حال اجرا باشند این message در فایل var/log/messages/ نوشته می شود.

کامپایل ماژول های هسته

ماژول های هسته کمی متفاوت نسبت به برنامه های معمولی کامپایل می شوند. برای اینکه بتوانید یک ماژول هسته را به درستی کامپایل کنید، نیاز به تنظیمات بسیار زیادی دارید. با پیچیده تر شدن ماژول ها این تنظیمات پیچیده تر می شوند.

خوشبختانه مکانیزمی به نام kbuild وجود دارد که تمام این تنظیمات را انجام می دهد. برای اگاهی بیشتر از این مکانیزم به فایلهای مستند هسته که در آدرس linux/Documentation/kbuild/modules.txt کد منبع هسته موجود است مراجعه کنید. برای اینکه بتوانیم از مکانیزم kbuild استفاده کنیم، بایستی Makefile ای با استاندارد آن بنویسیم.

برای این کار یک فایل به نام:

Makefile باز کرده و دستورات زیر را در آن بنویسید

```
obj-m += hello-1.oall: make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modulesclean: make -C
/lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

حال با اجرای دستور make ماژول خود را کامپایل کنید. در هسته ۲,۶ به بعد از پسوند ko برای نامیدن ماژول های هسته استفاده شده است که به راحتی قابل تمییز از o که پسوند فایل های object است می باشد.

برای بدست آوردن اطلاعاتی از ماژول خود دستور زیر را اجرا کنید:

```
# modinfo hello-1.ko
```

برای وارد کردن ماژول خود در هسته از دستور زیر استفاده کنید:

```
# insmod./hello-1.ko
```

اگر بعد از اجرای این دستور فایل var/log/messages/ را باز کرده و به انتهای آن بروید، خواهید دید که ماژول hello-1 در هسته load شده است. با دستور lsmod نیز ماژول load شده را خواهید دید. برای unload یا خارج کردن ماژول خود از هسته از دستور rmmod به صورت زیر استفاده کنید:

```
# rmmod hello-1
```

دوباره اگر فایل var/log/messages/ را باز کنید و به انتهای آن بروید خواهید دید که ماژول hello-1 از هسته خارج شده است.

نتیجه گیری

در این مقاله کرنل سیستم عامل لیوکس و عملکرد ماژول ها را بررسی کردیم. یکی از بهترین سیستم عامل های که میتوان از آن به عنوان جایگزین ویندوز و یا مکینتاش استفاده نمود Linux است. سیستم عامل لینوکس به شما این قابلیت

را میدهد که هر چه بخواهید با آن انجام دهید چرا که کاملا Open Source بوده و دست شما را در ویرایش کد ها باز گذاشته است. لینوکس دارای Distribution های فراوانی میباشد که هر کدام از آنها به هدف مشخصی ساخته شده اند. سیستم عامل Linux یک سیستم عامل OpenSource (متن باز و رایگان) می باشد که در کنار ابزارهایی که GNU برای آن تولید کرده است تبدیل به یک سیستم عامل کامل می گردد به همین دلیل معمولا GNU/Linux نامیده می شود. از نظر فنی سیستم عامل لینوکس را می توان نمونه متن باز سیستم عامل یونیکس نامید که بسیاری از ویژگی های مثبت این سیستم عامل را نیز به ارث برده است.

کرنل بخش اصلی و مرکزی سیستم عامل لینوکس (هر سیستم عاملی) است که پردازنده، RAM و دستگاه های جانبی را مدیریت می کند. Kernel پایین ترین سطح یک سیستم عامل است.

هسته لینوکس یک هسته سیستم‌عامل است که با کمک توسعه‌دهندگان در سراسر جهان پیشرفت داده شد است هسته لینوکس، آزاد و متن باز یکپارچه، ماژولار (modular) و شبه یونیکس و بسیار قابل تنظیم است. هسته لینوکس بر روی طیف گسترده‌ای از سیستم‌های محاسباتی، مانند سیستم‌های توکار، دستگاه‌های تلفن همراه (از جمله استفاده از آن در سیستم عامل اندروید)، رایانه‌های شخصی، سرورها، حافظه‌های اصلی و ابر رایانه‌ها مستقر شده‌است.

چه می شود اگر ویندوز درایورهای موجود را از قبل نصب کرده باشد و شما فقط نیاز باشد درایورهای مورد نیاز خود را فعال کنید؟ این در واقع همان کاری است که ماژول های هسته برای لینوکس انجام می دهند. ماژول های هسته که به عنوان یک ماژول هسته قابل بارگذاری (LKM) نیز شناخته می شوند، برای حفظ عملکرد هسته با تمام سخت افزارهای شما بدون مصرف تمام حافظه موجود شما در دسترس هستند.

ماژول به طور معمول برای مواردی مانند دستگاه ها، سیستم فایل ها و درخواست های سیستم را به هسته اصلی می افزاید. LKM ها دارای پسوند.ko هستند و به طور معمول در دایرکتوری / lib / modules ذخیره می شوند. به دلیل ماهیت مدولار آنها به راحتی می توانید هسته خود را با تنظیم ماژول ها برای بارگذاری، یا بارگیری نکردن، هنگام راه اندازی با دستور menuconfig یا با ویرایش پرونده / boot / config خود تنظیم کنید، یا می توانید ماژول های موجود در بوت را با modprobe لود و آنلود کنید.

منابع

گلین مودی: برنامه یاغی: جنبش لینوکس و بازمتن، انتشارات پرسیوس، شابک ۳-۹۹۵۲۰-۷۱۳-۰ جیددا آر (۲۰۰۴)

http:/linuxreview, org.../
Remo Suppi Boldrito, "The GNULinux Operating System", 2009... .
Linus Torvalds, "Linux: a Portable Operating Thesis...
UNIVERSITY OF HELSINKI Department of Computer Science, 1997... .
Ellen Siever, " Linux in a Nutshell, 6th Edition... ,"
DevynCJohns _ "Linux Kernel", 2014... .
Brian Hackett, "TYPE SAFETY I THE LINUX KERNEL" PHD Thesis...
Do cumentation" _ Master thesis: Comenius Univers ity, 2006... _
Clustering", Master thesis P O LYTECHNIC UNIVERSITY OF VALENCIA Faculty...
Mehedi Al Mamun, " OPERATING SYSTEM:S SECURITY: LINUX", A...

Brian Hackett, " TYPE SAFETY IN THE LINUX KERNEL" PHD...
Bitzer, "Commercial _ open source software: The role of...
Machtelt Garrels, " Introduction to Linux" A Hands On Guide... ,

The kernel of the Linux operating system and the performance of the modules

Farshid Sahba 1
Ahmad Tafazzoli 2
Farzaneh Farazandeh 3

Date of Receipt: 2021/05/15 Date of Issue: 2021/05/26

Abstract

Today, when the Internet plays an important role in the exchange of information and everyone is thinking about getting the latest information, we should also try to make the most of up-to-date science. One of these sciences is the Linux operating system. Linux, which here refers to the Linux kernel, is the most widely used software open source software in the history of computer software and is present wherever you can think of; On computers, servers, cell phones, televisions, submarines, airplanes, self-driving cars, and even on the International Space Station. But perhaps few people know what a kernel really is and how it works? What is meant by operating system kernel and what does the Linux kernel include? What is a Linux kernel module? What are the types of modules? And how do they get to the core? These are all questions that will be addressed in this study.

Keywords

Operating system, kernel, Linux, module, Linux kernel, kernel

1. Assistant Professor, Department of Information Technology Engineering, Ghiasuddin Jamshid Kashani University, Abyek, Iran.
2. Master student, Department of Information Technology Engineering, Ghiasuddin Jamshid Kashani University, Abyek, Iran.
3. Master student, Department of Information Technology Engineering, Ghiasuddin Jamshid Kashani University, Abyek, Iran.